TR-86- 2

AN EXPERT SYSTEM FOR CONTROL AND SIGNAL PROCESSING WITH AUTOMATIC FORTRAN PROGRAM GENERATION

by

P. Chancelier, C. Gomez, J.P. Quadrat,
A. Sulem, G.L. Blankenship, A. LaVigna,
D.C. MacEnany and I. Yan

Proc. MTNS Symposium, Stockholm 1986

AN EXPERT SYSTEM FOR CONTROL AND SIGNAL PROCESSING WITH AUTOMATIC FORTRAN PROGRAM GENERATION

P. Chancelier,† C. Gomez,†
J.P. Quadrat,† A. Sulem†

G.L. Blankenship,†† A. LaVigna,††
D.C. MacEnany,†† I. Yan††

Abstract: A prototype expert system for the treatment of stochastic control and nonlinear signal processing problems is described with several illustrative examples. The system is written in MACSYMA, LISP, and PROLOG. It accepts user input in natural language or symbolic form; it carries out the basic analysis of the user's problem in symbolic form (e.g., computing the Bellman dynamic programming equation for stochastic control problems or the Zakai equation and the estimation Lie algebra or likelihood ratio for nonlinear filtering problems); and it produces output in the form of automatically generated FORTRAN code for the final numerical reduction of the problem. The system also has a module using PROLOG which can check the well-posedness (existence and uniqueness) of certain classes of linear and nonlinear partial differential equations specified in symbolic form by computing a natural Sobolev space for the solutions and verifying classical existence and uniqueness criteria for the given equation using MACSYMA for the computations and PROLOG for the logical analysis. Sample sessions with three of the modules of the system are presented to illustrate its operation. The status of the system and plans for its further development are described.

1. Introduction

In this paper we shall describe a prototype *expert system* using symbolic manipulation languages developed for the analysis and design of stochastic control and signal processing systems. The system has four major components: (i) a modular system of programs written in MACSYMA[1] which "solve" certain stochastic control and filtering problems in symbolic form; (ii) an "intelligent" command language interface using OBLOGIS, a PROLOG system written in LISP [1]; (iii) a "theorem proving module," also using OBLOGIS, capable of checking the well-posedness (existence and uniqueness of solutions on a Sobolev space determined by the module) of certain classes of linear and nonlinear PDE's in symbolic form; and (iv) a general purpose module for generating

†INRIA, Domaine de Voluceau, Rocquencourt B.P. 105, 78153 LE CHESNAY CEDEX, FRANCE.

†† Electrical Engineering Department, University of Maryland, College Park, Maryland 20742, USA. The research of G.L. Blankenship was supported in part by NSF Grant DMC 85-02101. The research of A. LaVigna and D. MacEnany was supported in part by ONR Graduate Fellowships. The research of I. Yan was supported in part by a Graduate Fellowship from the Systems Research Center (NSF Grant OIR-85-00108) at the University of Maryland. Address correspondence concerning this paper to G.L. Blankenship or J.P. Quadrat.

[1]MACSYMA is a language for symbolic manipulation developed at Project MAC at MIT. MACSYMA is a trademark of *Symbolics, Inc.*

FORTRAN programs from the symbolic manipulation modules of the system. The function of the system is to accept input from the user in natural language with model equations expressed in symbolic form, to "expertly" (automatically) select a solution technique for his control or filtering problem, to reduce his model equation by symbolic manipulations to a form appropriate for the technique, checking well-posedness of the model along the way; and to automatically generate a numerical language (FORTRAN) program realizing the solution algorithm.

In this way the system serves as a kind of **symbolic compiler** for control systems engineering. This is its intended purpose. See Figure 1.

The system which now exists is capable of treating, among other problems[2], stochastic optimal control problems by symbolically evaluating the Hamilton-Jacobi

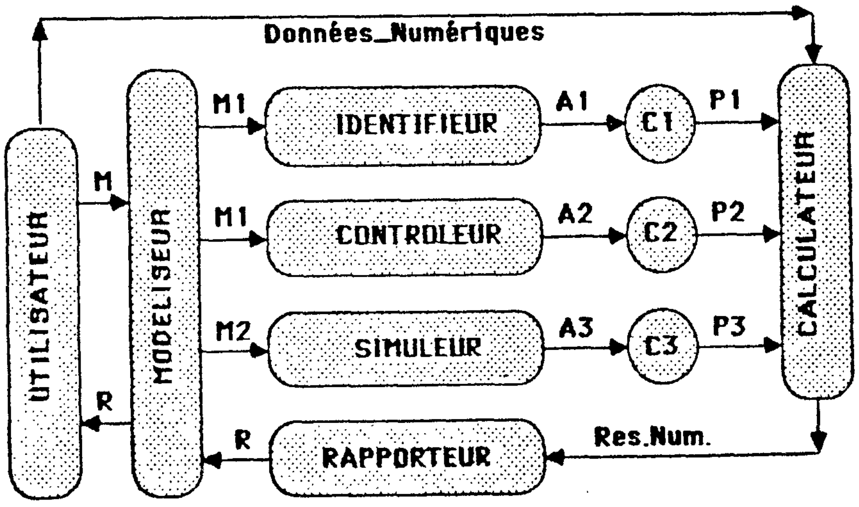


Figure 1. Structure of the expert system.

[2]It can also handle stochastic approximation, decentralized stochastic control, optimal control by the Maximum Principle, system identification, and certain nonlinear control and filtering problems. Several other modules are under development, including a program for automatic implementation of the Hunt-Su-Meyer method for exact linearization of nonlinear control systems [2] (suggested by C.I. Byrnes).

dynamic programming equation for the evolution of the optimal return function, selecting from four methods for the analysis, and then producing a FORTRAN code for the numerical solution of this equation [3]. To execute the system, one inputs[3] the system model equations, identifying state and control variables, and specifies in general terms the type of optimization problem involved. The cost function, the initial conditions, and any other boundary conditions are also entered. The MACSYMA portion of the code carries out the calculations involved in implementing dynamic programming for the optimization problem. That is, it carries out, in symbolic form, the differentiations, function minimizations, and routine algebra necessary to compute the Hamilton-Jacobi equation which describes the evolution of the optimal cost and the functional form of the optimal control law. This portion of the code involves a grammar based on MACSYMA expressions. A subsequent portion of the code which is written in MACSYMA and LISP evaluates FORTRAN expressions for the appropriate MACSYMA expressions and produces "dimension", "write", "format", "goto", "logical if", "numerical if", and other standard FORTRAN lines for the construction of a FORTRAN program. The FORTRAN (subroutine) is written into a file using a MACSYMA command. For typical stochastic control problems execution of the MACSYMA program and generation of the FORTRAN code takes about 30 seconds. The programmer can at this point enter numerical values for the system functions and data and then execute the FORTRAN code. (These could be also entered automatically from preassigned files, and the FORTRAN code executed automatically.) A sample session with this component of the system is summarized in the next section.

The advantages of this system are clear. Given a new control problem, the time involved in analyzing the dynamic programming equation and then writing the FORTRAN code to execute it is eliminated. The mistakes and the time required to test and debug the FORTRAN code are also eliminated. This is a major advantage of the system in its present form. Most importantly, the system allows the engineer to interact with the computer for design in terms of symbolic expressions. In this way he can modify his analysis or design problem by modifying the symbolic functional form of the model. The FORTRAN subroutines that he might have to modify by hand to accomplish this in conventional design procedures are written automatically for him. The advantages of working at the higher level are clear. As we shall show later, the system may also be used for basic research in control theory.

The stochastic control portion of the system also has *production rules* written in OBLOGIS which take advantage of the of the special structures associated with such problems to select or guide the selection of analytical or numerical procedures to treat specific applications.

1.1 Related work

Several systems for computer-aided-design of control systems have been developed recently:

(1) ORACLS by NASA [4];

[3]Using a special grammar of MACSYMA expressions, or a natural language interface. See section 2.1.

(2) DELIGHT.MIMO by the University of California, Berkeley and Imperial College, London [5];

(3) DELIGHT.Maryland by the University of Maryland [6];

(4) CACE-III by the General Electric Company and RPI [7];

(5) The Federated System by the General Electric Company [8];

(6) and various commerical packages like Matrix$_X$, Program CC, and Cntl-C.

In addition, there has been some excellent work on general and specific algorithms for numerical problems associated with control system design [9]. The cumulative product of these efforts is a strong library of *numerical* systems and packages for the analysis and design of numerically defined control systems.

The system in [7] is closest to ours in spirit. It has an interface written in LISP which provides an "intelligent" interaction with pre-existing design packages.

The system described here has a different function. It is designed to automatically generate numerical programs for the solution of problems posed in symbolic form. By using symbolic manipulation programs, the system permits design efforts to take place at the "next level up" from FORTRAN code. It interfaces more naturally with AI constructs since it is based in LISP and involves symbolic expressions, as opposed to the numerical structures of FORTRAN. In this way it facilitates the incorporation of an intelligent natural language interface. It would be difficult to design such an interface in FORTRAN, and it would be equally awkward to write theorem proving modules in FORTRAN.

1.2 Summary

In the next section we present a summary of a sample session with one module of the system with the interaction taking place through the natural language interface. In the Section 3 we explain how the MACSYMA portion of the code is developed for one particular application - the construction of approximate nonlinear filters for systems with small parameters. In Section 4 we present a sample session with the PROOFONPDE module. In Section 5 we discuss the present status of the overall system and plans for its further development.

Acknowledgements: We would like to thank C.I. Byrnes, P. Kumar and K. Paul for their contributions to the University of Maryland portion of this work.

2. A sample session for optimal stochastic control

The operation of the system in treating a static, elliptic stochastic optimal control problem is illustrated in the transcript which follows. The problem is first specified using symbolic expressions. In the interaction, the user specifies the states, controls, and system structure, including the physical interpretation of the problem. In this case, the system is told that the problem corresponds to the problem of optimal production control of a hydropower system. The states are the volumes of water stored for release in two reservoirs, and the controls are the release rates. The system retains this information, and returns it when asked for the physical meaning of the problem. Notice that the system returns a summary of the information it has on the mathematical structure of the problem at any stage, and the information is returned in the same terms used to introduce it. Notice too that the system checks for well-posedness of the problem by keeping track of dimensions, indicating that the problem type (parabolic or static) has not yet been declared, and asking for the specification of a discount factor. Finally, and most importantly, notice that the information the user has to enter corresponds to the information necessary to *write down* a dynamic programming problem. The system performs all the required calculations needed to determine the Belman equation from the system dynamics and derive the numerical implementation of the equation.

When the problem specification is complete, as shown in the example, the system invokes a MACSYMA routine, in a procedure which is invisible to the user, to compute the associated Hamilton-Jacobi-Bellman (HJB) dynamic programming equation.[4] The system will display the HJB equation in symbolic form if asked. At this point, the user asks for the generation of a FORTRAN program to solve the HJB equation numerically. This may be done in a variety of ways. The user may simply enter the command "graph," which is equivalent to asking the system to compute the optimal return function and make a graph of it. The system generates a FORTRAN program and tells the user where to find it; it then proceeds to execute the FORTRAN code and produce the desired graph (shaded by hand in the transcript). The numerial results are stored in a file, and this is also reported to the user.

2.1 Interaction through the command language interface

In the session (Figures 2a,b) the user sets up the stochastic control problem

$$\inf \{ \int_0^{T_s} f\,(x_1(t), x_2(t), u_1(t), u_2(t))\, e^{-qt}\, dt \;\mid\; u_1, u_2 \}$$

$$f\,(u_1, u_2, x_1, x_2) = u_1^2 + u_2^2 + x_1(1-x_1)x_2(1-x_2)$$

$$+ \; ((\frac{1}{2} - x_1)x_2(1-x_2))^2 + ((\frac{1}{2} - x_2)x_1(1-x_1))^2$$

with the discount rate $q = 1$ and the state equations

$$dx_1(t) = [\,\frac{1}{2} - u_1\,]\,dt + dw_1(t)$$

$$dx_2(t) = [\,\frac{1}{2} - u_2\,]\,dt + dw_2(t)$$

$$(u_1(t), u_2(t)) \in [0,1]^2, \quad (x_1(t), s_2(t)) \in [0,1]^2.$$

as a model for the control of a pair of hydropower reservoirs with states x_1 and x_2, the stored water volumes, and controls u_1 and u_2, the release rates of the water from the

[4]In fact, the system is capable of selecting from several methods for treating a given optimization problem. See section 2.4.

reservoirs. The cost function corresponds, in effect, to the cost of producing the equivalent thermal power. See, e.g., [10]. The processes $w_1(t)$ and $w_2(t)$ are independent standard Wiener processes, and τ_s is the first exit time of the state from the domain $D = [0,1]^2$.

The optimality conditions for this problem lead to the following HJB equation

$$-v + \min\{ (u_1,u_2) \in [0,1]^2 \mid u_1\frac{\partial v}{\partial x_1} + u_2\frac{\partial v}{\partial x_2}$$

$$+ u_2^2 + u_1^2 + (0.5-x_1)(1-x_2)^2x_2 + (1-x_1)x_1(1-x_2)x_2$$

$$+ (1-x_1)^2x_1(0.5-x_2) + \frac{\partial^2 v}{\partial x_1^2} + \frac{\partial^2 v}{\partial x_2^2}\}$$

on the domain $D = \{ (x_1,x_2) \in [0,1]^2 \}$ with the boundary conditions

$$v = x_1x_2 \text{ for } x_1 = 0 \text{ and } x_2 = 0$$

$$v = (1-x_1)(1-x_2) \text{ for } x_1 = 1 \text{ and } x_2 = 1$$

In the listing user input follows the arrow (==>). The system's response is on the lines immediately thereafter. Note the role of the help facility and the responses of the system to an incompletely posed problem. The total time required for the session was about 30 seconds on the Honeywell Multics system at INRIA.

EXEMPLE DE SESSION

```
        hello
        we shall try to solve your stochastic
control problem
        do you want to keep a previous statement?
        no
        do you want to keep the following session?
        no
        please give the statement of the problem
        if you have trouble type help

        ==>[x1,x2] is a state variable denoting
 the water-levels
         state : [x1, x2]

       physical meaning of [x1, x2] : water - levels

       ==>[u1,u2] is a control variable
denotingturbined-water
       control : [u1, u2]
       physical meaning of [u1, u2] : turbined -
-water

       ==>g =0.5 is the drift
       the dimensions of the state and the drift
must be the same

       ==>h = [0.5-u1,0.5-u2] is a drift
       drift : h
       value of h : [0.5 - u1,         0.5 - u2]

       ==>what is h
       drift : h
       value of h : [0.5 - u1,         0.5 - u2]

        ==>s = [0.5,0.5] is the diffusion
 diagonal matrix
        diffusion : s
        value of s : [0.5,0.5]


       ==>c
=u1^2+u2^2+x1*(1-x1)*x2*(1-x2)+
 ((0.5-x1)*x2*(1-x2))^2+
 ((0.5-x2)*x1*(1-x1))^2  is the instantaneous
 cost
       instantaneous : c

       value of c : (0.5 - x1)(1 - x2)^2 x2

 +(1-x1)x1(1 - x2) x2+ (1 - x1)^2 x1(0.5 - x2)
 +u2^2+u1^2
```

```
       ==>what do you know
       instantaneous : c
       value of c : (0.5 - x1)(1 - x2)^2 x2
       + (1 - x1) x1 (1 - x2) x2
       + (1 - x1)^2 x1(0.5 - x2)+ u2^2+ u1^2
       diffusion : s
       value of s : [0.5,0.5]
       drift : h
       value of h : [0.5 - u1,0.5 - u2]
       control : [u1, u2]
       physical meaning of [u1, u2] : turbined -
-water
       state : [x1, x2]
       physical meaning of [x1, x2] : water - levels

       ==>the problem is static
       the problem is static

       ==>dynammic programming equation
       a discount factor is missing
       do you wish to stop ?
       no

       ==>q = 1 is a discount factor
       discount : q
       value of q : 1

       ==>help
       do you need help to

       - give the dynamic of the problem (1)
       - give the horizon and the limit conditions
(2)
       - generate the bellman equation (3)
       - generate the fortran program and
numerical results (4)
       - ask for the edition functions of the data
base of the robot (5)
       please answer the corresponding number
       2
       you must precise if the problem is static or
time-dependent by typing (the problem is)
static/parabolic
       when the problem is static you must give
the discount factor,when the problem
       is parabolic you must give the horizon and
the final cost
       more help ?
       yes
       examples:x = 1 is a discount (factor)
       tt = 1 is the horizon (time)
       f = 2 is a final cost
       if for instance the limit condition at
        x = 0 is of the neuman/dirichlet type
       and equal to x+1 you type:
       x = 0 dirichlet/neuman 1+x
       you can substitute stopping to dirichlet and
 reflexion to neuman
```

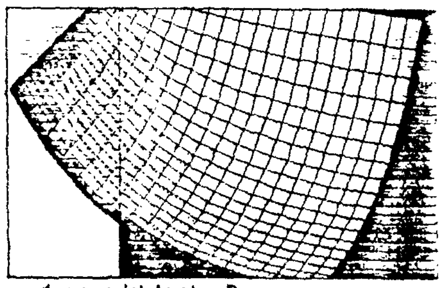
Figure 2a. A session with the system for optimal stochastic control.

```
==>x1 =0 dirichlet 2
limit condition for x1 =0 : 1
dirichlet type

==>delete limit condition x1 = 0
limit condition deleted

==>x1 =0 dirichlet 1
limit condition for x1 =0 : 1
dirichlet type

==>x2 =0 dirichlet 1
limit condition for x2 =0 : 1
dirichlet type

==>x1 =1 neuman 0
limit condition for x1 =1 : 0
neuman type
limit condition for x1 =0 : 1
dirichlet type

==>x2 =1 neuman 0
limit condition for x2 =1 : 0
neuman type
limit condition for x2 =0 : 1
dirichlet type

==>equation
hamilton-jacobi equation :
```

$$\min_{\{u1,u2\}}\Big((0.5-x1)(1-x2)^2x2 + (1-x1)x1(1-x2)x2 + (1-x1)^2x1(0.5-x2)+u_1{}^2+u_2{}^2 + (0.5-u2)\frac{dv}{dx_1} + (0.5-u1)\frac{dv}{dx_2}\Big) + 0.5\frac{d^2v}{dx_1{}^2} + 0.5\frac{d^2v}{dx_2{}^2} - v = 0$$

$$v = 1 \text{ for } x1 = 0$$
$$v = 1 \text{ for } x2 = 0$$
$$\frac{dv}{dx1} = 0 \text{ for } x1 = 1$$
$$\frac{dv}{dx2} = 0 \text{ for } x2 = 1$$

```
        do you wish to stop ?
        no

==>generate

        you will find the subroutine in the segment
belman.fortran
        do you wish to stop ?
        no

        ==>principal program

        you will find the principal program in the
segment pp.fortran
        do you wish to stop ?
        no

        ==>execute

        you will find the fortran program in the
segment num.fortran
        you will find the numerical results in file08
where v denotes the bellman
        function and u the optimal control
        do you wish to stop ?
        no

        ==>graph
```



```
        do you wish to stop ?
        no
```

Figure 2b. Continuation of the session.

2.2 The MACSYMA grammar for the example

The class of stochastic control problems treated by the system is defined internally by a special grammar shown in Figure 3. The grammar uses the Backus-Naur notation where "|" stands for "or" and <word> stands for a nonterminal word. There are also semantic constraints which define a sublanguage of this grammar. These constraints effect the use of <y>. We shall not discuss this feature here.

$$< \mathit{stochastic-control-problem} > ::= < \mathit{domain} >, < \mathit{inside-condition} >, < \mathit{boundary-condition} >$$

$$< \mathit{domain} > ::= [0,1]^n \ | \ [0,1]^n \times [0,T].$$

$$< \mathit{boundary-conditions} > ::= \sum_{< \mathit{boundary-element} >} < \mathit{boundary-conditions} >, < \mathit{boundary-element} >$$

$$< \mathit{boundary-element} > ::= \{< y >: < y > \in < \mathit{domain} >, < x_i > = 1\} \ |$$

$$\{< y >, < y > \in < \mathit{domain} >, < x_i > = 0 \ |$$

$$\{(x,1), x \in [0,1]^n\}$$

$$< x_i > ::= x_1 \ | \ x_2 \ | \ \cdots \ | \ x_n$$

$$< y > ::= x \ | \ (x,t)$$

$$< \mathit{boundary-condition} > ::= V = f \ | \ \frac{\partial V}{\partial n} = f$$

$$< \mathit{operator} > ::= < \mathit{evolution-operator} > + < \mathit{space-operator} >$$

$$< \mathit{evolution-operator} > ::= \ldots | \ \frac{\partial V}{\partial t}$$

$$< \mathit{operators} > ::= < \mathit{operator} > \ | \ < \mathit{operator} > + < \mathit{operators} >$$

$$| \ \min \{< \mathit{operator} >, < \mathit{operators} >\}$$

$$< \mathit{inside-condition} > ::= < \mathit{operators} > = 0$$

$$< \mathit{space-operator} > ::= -\lambda(< y >)\, V + \sum_{i=1}^{n} b_i(< y >) \frac{\partial V}{\partial x_i} + \sum_i a_i(< y >) \frac{\partial^2 V}{\partial x_i^2} + C(< y >)$$

$$| \ \min \{u \in \mathbb{R}^m : -\lambda(< y >), u)V + \sum_{i=1}^{n} [b_i(< y >, u) \frac{\partial V}{\partial x_i}$$

$$+ a_i(< y >, u) \frac{\partial^2 V}{\partial x_i^2}] + C(< y >, u)\}$$

Figure 3. Prototype grammar for stochastic control.

The MACSYMA program which sets up the HJB equation in the example, written in the special grammar for stochastic control problems, is shown in Figure 4. The grammar is highly efficient, and so the MACSYMA program is quite compact.

2.3 Specialized natural language interface

The interface illustrated in the interactive session is a program written in LOGIS [11] which recognizes the specialized terminology of stochastic control. The sentences the user enters are analyzed and matched against aptterns of typical sentences. Depending on the type of sentence, various actions are initiated: answer the user; access a data base; etc.

For example, when the user enters

let [x_1, x_2] be a state variable denoting water level

the corresponding internal pattern is:

?- let !x !- !- !type ?- denoting !- !physical-meaning

where ?- matches anything, !- matches one word, and !x matches one word and binds it

```
appel():=(
    fff:x1*(1-x1)*x2*(1-x2),
    fsc:((0.5-x1)*x2*(1-x2))**2+((0.5-x2)*x1*(1-x1))**2+
      2*x2*(1-x2)+2*x1*(1-x1),
    condi:[[dirich,x1*x2],[dirich,x1*x2]],
    condim:[[dirich,(1-x1)*(1-x2)],[dirich,(1-x1)*(1-x2)]],
    type:[2,stat,ecriture,pasmoy,condlim,condi,condim,elp,1,difu,derive,
      plus,dif,[1,1],belm,2,newto,an,u1*p1+
      u2*p2+u1**2+u2**2+fsc+fff,param,[]],
    hjb(type,prodyn,belman));
```

Figure 4. MACSYMA program for stochastic control

to x. The result of the pattern matching is an association list:

(x . [x_1, x_2], type . state, physical-meaning . water level)

The associated actions are:

- transform the data and store them
- verify the properties and meaning of the data
- answer the user.

The answer is:

state: [x_1, x_2]

physical meaning of [x_1, x_2]: water level

Through the interface the user can delete a variable, ask for the meaning of a variable, and ask for all the data and variables in the system at any point in the interaction. The user can request certain actions including the generation of the dynamic programming equation associated with a problem, or the generation of FORTRAN code to implement a specific algorithm for the optimization problem.

2.4 Rules for selection of solution method

Using the logic programming facilities of the OBLOGIS system, it is possible to develop rules for evaluating the problem statements provided by the user to test whether or not they are complete and to request additional information. In effect, the system is able to determine when an optimal stochastic control problem is completely posed.[5] Rules have also been incorporated into the system to select among four different methods for treating optimal stochastic control problems. The MACSYMA portion of the program is capable of treating stochastic optimal control problems using four different methods: (i) direct numerical solution of the Bellman equation for systems with small state dimension using methods in [12,13,14]; (ii) a decoupling method for systems which have decoupled dynamics and product form objective function [15]; (iii) the stochastic gradient method [16]; and (iv) a regular perturbation method [17,18].

The problem statement supplied by the user is tested to determine which method is applicable. For example, the (regular) perturbation method is applicable if

- the problem is a control problem
- the time horizon is finite
- the noise intensity is small
- there are no state constraints
- there are not control contraints

[5]See section 4 where the issue of ***well-posedness*** is discussed.

- the second order differential operator in the Bellman equation is regular

These conditions are realized in the following LOGIS PROLOG clauses:

```
(← (methode !perturbation)
   (type !commande)
   (horizon !fini)
   (petit-bruit)
   (pas-constrainte !etat)
   (pas-constraint !commande)
   (non-degenere !hamiltonien !commande))
```

Here "←" signifies installation of a clause and "!" introduces a PROLOG constant. The first list in the clause is the conclusion and the subsequent lists are the hypotheses which must be tested to satisfy the conclusion. This order reflects the backward chaining inference structure of PROLOG.

2.5 FORTRAN program output

Numerical evaluation of the solution of the Bellman equation is done by a FORTRAN program written automatically by the system. A listing is given in the Appendix. Note that it is a fully commented, "stand alone" program written in FORTRAN IV. It contains some 120 lines, including comments. It is produced in 1-2 seconds by the system in the course of the interactive session.

3. Algebraic and asymptotic simplification of nonlinear filters

The problem of estimating one diffusion Markov process $\{ x_t \}$ in terms of observations of another $Y_t = \sigma\{ y_s , \ s \leq t \}$ based on the model (Ito calculus)

$$dx_t = f(x_t)dt + g(x_t)dw_t \qquad (1)$$

$$dy_t = h(x_t)dt + du_t , \quad 0 \leq t \leq T < \infty$$

is fundamental in many engineering applications. Here f, g, and h are smooth functions, w_t, u_t are independent standard Wiener processes, and x_0 is a random variable independent of (w_t , u_t) for all t. The problem of recursively estimating x_t given the σ-algebra Y_t generated by the observations, may be formulated in terms of the (Stratonovich) stochastic partial differential equation for the unnormalized conditional density

$$du(t,x) = (L^* u)(t,x)dt + h(x)u(t,x)dy_t$$

$$L^* u = \frac{1}{2}\partial_{xx}(g^2(x)\,u) - \partial_x(f(x)\,u) - \frac{1}{2}h^2(x)\,u \tag{2}$$

$$u(0,x) = p_0(x), \text{ the density of } x_0$$

That is, if (2) has a nice solution, then the conditional density of x_t given Y_t is

$$p(t,x) = u(t,x)/[\int u(t,z)dz].$$

A considerable effort has been devoted to the search for (recursive) finite dimensional "representations" of estimators in terms of various "sufficient statistics" of either the solution of (2) or other conditional statistics (such as the conditional mean). This effort, which has produced few such estimators, has nevertheless increased our understanding of the system (2) and of the tools available for its treatment [19]. In [20] a systematic procedure was developed for constructing approximations to the solutions of (2) in terms of certain Lie algebras of differential operators associated with the equation. See also [21,22,23]. The procedure is based on the presence of a small parameter in the model (1), and it uses the methods of asymptotic analysis to derive the approximations.

3.1 Asymptotic analysis and the estimation algebra

The idea is as follows: Consider the simple perturbation of the Kalman filtering problem treated in [20].

$$dx_t = ax_t\,dt + dw_t$$

$$dy_t^\epsilon = [\,x_t + \epsilon(x_t)^k\,]dt + du_t \tag{3}$$

$$y_0^\epsilon = 0, \quad x_0 \text{ has Gaussian density } p_0(x), \ k \geq 2.$$

The conditional density of x_t^ϵ given Y_t^ϵ satisfies

$$du^\epsilon(t,x) = [L^* - \frac{1}{2}(x + \epsilon x^k)^2]u^\epsilon(t,x)dt + (x + \epsilon\, x^k)\,u^\epsilon(t,x)\,dy_t^\epsilon \tag{4}$$

$$u^\epsilon(0,x) = p_0(x)$$

$$L^* u = \frac{1}{2}\partial_{xx}u - a\,\partial_x(xu)\,.$$

Assume that u^ϵ has an asymptotic expansion

$$u^\epsilon = u_0 + \epsilon u_1 + \epsilon^2 u_2 + \cdots \tag{5}$$

Substituting this into (4) and equating coefficients of like powers of ϵ gives

$$du_0 = L^* u_0\, dt + x\, u_0\, dy_t^{\epsilon} \tag{6}$$

$$u_0 = p_o(x)$$

$$du_k = L^* u_k\, dt + x\, u_k\, dy^{\epsilon}_t + x^k u_{k-1}^{\epsilon_t}$$

$$u_k(0,x) = 0, \quad k = 1,2,\ldots$$

Defining the vector

$$U_n(t,x) = \begin{bmatrix} u_0 \\ u_1 \\ \cdots \\ u_n \end{bmatrix} \tag{7}$$

we have

$$dU_n(t,x) = [(L^* - \frac{1}{2}x^2)E_1 - x^{k+1}E_2 - \frac{1}{2}x^{2k}E_3]\, U_n(t,x)\, dt \tag{8}$$

$$+ [xE_1 + x^k E_2]\, U_n(t,x)\, dy_t^{\epsilon}$$

where the $(n+1)$ x (n +1) matrix E_i has zero entries everywhere except for $1'$ s on the subdiagonal $(i,1)(i+1,2),\ldots,(n+1,n+2-i)$ (i.e., on a particular subdiagonal). Note that E_1 is the identity matrix.

The equation for u_0 is the Kalman conditional density equation (given y_s^{ϵ}, $s \leq t$). Therefore, it has a closed form analytical solution - a Gaussian density. This density defines the Green's function for all the subsequent equations, $k = 1,2,\ldots$, in (6). Since u_1 involves this Green's function and $x^k u_0\, dy_t^{\epsilon}$ as a forcing function, u_1 can also be computed explicitly. It follows that all the terms $u_1, u_2, \cdots$ can be computed explicitly; however, the complexity of the calculation increases very rapidly. By computing $U_n(t,x)$ we produce an approximation to the conditional density which is accurate to $O(\epsilon^{n+1})$.

All these calculations can be done automatically in symbolic form by MACSYMA. To do this, we use Lie theory. First, we rewrite (8) abstractly as

$$\dot{U}_n(t) = AU_n(t) + BU_n \dot{y}_t^{\epsilon}$$

$$A = [(L^* - \frac{1}{2}x^2)E_1 - x^{k+1}E_2 - \frac{1}{2}x^{2k}E_3] \tag{9}$$

$$B = [x\, E_1 + x^k\, E_2]$$

Note that A is a second order differential operator and B is multiplication by a matrix valued function. The Lie algebra generated by A and B consists of all linear combinations of A and B and their consecutive commutator products, e.g.,

$$[A, B] = AB - BA \tag{10}$$

and subsequent repeated products (taken as differential operators). This is always a finite dimensional *solvable*[6] (e.g., "lower triangular" structure) Lie algebra for the class of filtering problems with polynominal coefficients described above.

Consequently, we can use the Wei-Norman theory [24] to solve (9) globally in the form

$$U_n(t, x) = [\, e^{g_1(t)A_1} e^{g_2(t)A_2} \cdots e^{g_d(t)A_d} \,](x) \tag{11}$$

where $(A_1, A_2, \ldots, A_d)$ is a basis for the Lie algebra generated by A and B, and the $g_i(t)$ are defined by a lower triangular system of nonlinear (stochastic) ordinary differential equations. These equations must be solved numerically (in general). The $g_i(t)'\,s$ may be thought of as a finite dimensional family of (approximate) conditional statistics for the original nonlinear filtering problem.

3.2 Symbolic algorithm structure

To implement the approximate nonlinear filter using this methodology and MACSYMA, one must program the following six step procedure:

(M1) Compute the differential operators A and B in (9) in terms of the original functions in the filtering problem model (1), modulo the appropriate power of the small parameter ϵ, if one is present in the model.

(M2) Compute basis elements $A_1, A_2, \ldots, A_d$ for the Lie algebra generated by A and B .

(M3) Compute ODE's for the $g_1, \ldots, g_d$ functions in (11) by substituting (11) into (9), differentiating, and inverting the resulting matrix equation.

(F1) Solve the ODE's for the $g_i'\,s$ numerically using an appropriate numerical integration scheme for stochastic ODE's [25].

(M4) Compute the approximate conditional mean $\hat{x}_t^\epsilon$ (the best mean square estimator of x_t^ϵ given Y_t^ϵ) by integrating the conditional density.

(F2) Represent the conditional mean computed in (M4) in FORTRAN code in terms of the actual measurements $\{\, y_s^\epsilon,\ s \leq t \,\}$ as input data to the code.

The steps labeled (M1)-(M4) are done in the MACSYMA part of the code. The step (F1) is best carried out in FORTRAN; and so, the MACSYMA step (M3) which computes the symbolic ODE's has an auxiliary component which writes the FORTRAN code to numerically integrate the ODE's.

The result of step (F2) is the output of the system - that is, a FORTRAN code to numerically evaluate the best estimate of the state in terms of numerical measurement data. All the steps (M1) - (M4) and (F1) (F2) are carried out automatically after the system model (the functions f , g , and h in (2)) are specified.

[6]A Lie algebra is *solvable* if the derived series of ideals $L^{(0)} = L$; $L^{(n+1)} = [\, L^{(n)}, L^{(n)} \,]$, $n \geq 0$ is the trivial ideal $\{\, 0 \,\}$ for some n.

The system also includes a test for *well-posedness* which consists of checking relative growth conditions on the coefficients *f, g,* and *h* for existence and uniqueness of a (classical) solution to the *robust form* of the Zakai equation. The criteria used are those developed in the paper [26]. As is the case in the other components of the system, complicated analysis is required to setup the existence and uniqueness test. This analysis is done "automatically" by the program. Since the existence and uniqueness theory for nonlinear filtering problems is not complete, the program cannot give a definitive analysis of a large class of problems. Rather, it uses the classical existence and uniqueness theorems for parabolic partial differential equations as adapted to a class of filtering models with smooth coefficient functions. It is not unlikely that more sophisticated techniques could be accomodated by simple modifications of the program.

It is possible to have MACSYMA offer a choice of numerical integration procedures to be (automatically) coded for computation of the state estimate. Our system does not now have this capability. However, it does have a second method for computing the conditional density based on evaluation of the likelihood ratio using formulas for integration of stochastic function space integrals developed in [27,28]. The system also has the capability to test the well posedness of (robust form of the) Zakai equation. In conducting this test, using the theorems in [26], the system tests the coefficients in the Zakai equation to see if they are polynominals. If so, a simple test suffices for well-posedness. If not, then a more complex computation is executed. See [29] for details.

3.3 A sample session with the filtering module

The *weak quadratic sensor* treated in the paper [20] is one of the few examples in which the estimation algebra and the Wei-Norman series representation of the conditional density have been worked out by hand. It is necessary to have such examples when programming in MACSYMA to validate the program.

In the session shown in Figure 5 the program is asked to analyze the simple filtering problem

$$dx_t = dw_t \tag{12}$$

$$dy_t = x_t + \epsilon\, x_t^2 + du_t$$

The system asks the user to declare the system as scalar or vector and then to enter the functions f ,g , and h in a standard format. It then sets up the Zakai equation, identifies the two operators in the equation which will be used to generate the estimation algebra, and proceeds to find the representation of the conditional density to the appropriate power of ϵ, in the form of a Wei-Norman series. The system returns a basis for the estimation algebra (modulo the designated power of ϵ) and various intermediate expressions of interest, including the matrix of structure constants for the estimation algebra. Its "final result" is a set of stochastic ordinary differential equations for the functions appearing in the exponentials of the Wei-Norman series.

The final output of the program is a FORTRAN program integrating these equations and computing numerical expressions for the conditional mean. The FORTRAN program written by the system is given in the Appendix. The complexity of the expressions in this program effectively illustrates the value of a tool for the automatic generation of programs. It would be tedious to type this code, and difficult to debug it by hand. If a single parameter in the system model were changed, then the entire code might have to be rewritten. This would be time consuming, and a poor use of engineering talent. On the other hand, the code in the Appendix was generated in about 10 seconds on a VAX 11/785 under modest load conditions.

4. A Sample Session with the Theorem Proving Module

The system contains a module called *PROOFONPDE* for the automatic verification of well-posedness of certain classes of nonlinear PDE's. The module is constructed in LOGIS [11]. It analyzes the coercivity properties of a given PDE and the monotonicity properties of its nonlinear components on a Sobolev space which it constructs in the course of the analysis. The theorems of functional analysis used in the arguments are encoded in rules in the LOGIS PROLOG syntax. The system interfaces completely with MACSYMA; and MACSYMA functions performing symbolic calculations are used when necessary. The problems treated are in abstract form

$$Au = f \quad in \quad \Omega \subset \mathbb{R}^n \tag{13}$$

$$B_j u = g_j, \quad j = 0,1,\ldots,m-1 \quad on \quad the \quad boundary \quad \Gamma \text{ of } \Omega$$

where A and B_j are differential operators. The system verifies the existence of a solution to (13) and finds the function space V to which this solution belongs.

The function spaces used are Sobolev spaces which are represented internally as MACSYMA objects. For example, the space $W_0^{m,p}(\Omega)$ is handled in the following way: It is typed by the user in MACSYMA syntax as **space(w,m,p,0)** where **space()** is a MACSYMA function that generates the list

((space simp) ((p m 0)) nil)

which serves as the internal representation of the space. When the MACYSMA **disp** function operates on this list, it prints $W_0^{m,p}(\Omega)$.

The main components of the formal calculus (Green's function application, variational formulation, computation of canonical forms, etc.) are written in LISP. This is done since the main part of the work consists of manipulating lists (MACSYMA expressions), and it is most efficient to do this in LISP. When necessary, MACSYMA functions (*diff, expand,* etc.) are called. The programming technique is data driven programming. This facilitates the updating of the knowledge base.

The portion of the system using existence theorems and deciding what computations must be done to verify the hypotheses is encoded in Horn clauses with the PROLOG syntax of LOGIS [11]. This is the core of the system which determines what theorem and what method will be applied in a given case. For example, the Lax-Millgram theorem is encoded in the following clause:

```
(c4) start();

        The model for the stochastic system in Ito form is as follows

                        dx(t)= f(x(t))dt + g(x(t))dw(t)

                        dy(t)= h(x(t))dt + dv(t)

              w(t),v(t) are independent, standard Wiener processes

Is this the vector or scalar case?
s;

input f(x(t))
0;

input g(x(t))
1;

input h(x(t))
x+e*x^2;

                                    f = 0

                                    g = 1

                                         2
                                h = e x  + x

Please enter a number corresponding to the highest power of 'e' allowed
1;

                      The Zakai equation for this problem is

                                         2
                                        d u
                                        ---
                                   2      2
                          3     u x     dx                   2
          du = dt (- e u x  - ---- + ---) + dy (e u x  + u x)
                                2       2

                                              2
                                             d u
                                             ---
                                      2        2
                             3     u x       dx            2
            pdops = [- e u x  - ---- + ---, e u x  + u x]
                                    2        2

                         2
                        d u
                        ---
                  2       2
         3     u x      dx          2      du           du     2
basis = [- e u x  - ---- + ---, e u x , e -- x, u x, --, e d u, e u x, e --,
                  2       2                dx           dx    ---         dx
                                                                2
                                                              dx

                                                                      u, e u]
```

```
                  [ 0  2  0  0  0  0  0  0  1 ]
                  [                           ]
                  [ 1  0  0  0  1  0  0  0  0 ]
                  [                           ]
                  [ 0  0  0  1  0  0  0  0  0 ]
                  [                           ]
                  [ 3  0  1  0  0  0  0  0  0 ]
                  [                           ]
        tmatrix = [ 0  2  0  0  0  0  0  0  1 ]
                  [                           ]
                  [ 0  0  0  0  0  0  1  0  0 ]
                  [                           ]
                  [ 0  0  0  0  0  1  0  0  0 ]
                  [                           ]
                  [ 0  0  0  0  0  0  0  0  0 ]
                  [                           ]
                  [ 0  0  0  0  0  0  0  0  0 ]

              Exponentiating the transition matrix

                  Exponentiation completed


             The system of D.E.'s has been determined

diequations = matrix([1], [dy cosh(2 t) + dy cosh(t) - dy],

[2 dy sinh(t) - 2 dy sinh(2 t)], [dy cosh(t)], [- dy sinh(t)],

[dy cosh(2 t) - 2 dy cosh(t) + dy], [- 2 g  dy sinh(t) - g  dy cosh(t)],
                                          2                3

[2 g  dy cosh(2 t) - g  dy sinh(t) - 4 g  dy cosh(t) + 2 g  dy],
    4                 3                  4                  4

                                       2
[- g  dy sinh(t)], [- dy sinh(2 t) + g  dy cosh(2 t) + 2 g  g  dy sinh(t)
    4                                  4                    2  5

                                                  2                    2
 - g  g  dy sinh(t) + dy sinh(t) + g  g  dy cosh(t) - 2 g  dy cosh(t) + g  dy])
    3  4                              3  5                 4               4

do you want fortran generated? yes; or no;
yes;

input xo
1;

input eps
1;

To what file do you want the fortran sent?
demfort2;

(c5) exit();
```

Figure 5. A sample session with the nonlinear filtering module.

((solution space)
 (well defined)
 (linear $principal-op $solution)
 (elliptic $principal-op space)
 (continuous $principal-op space))

This is interpreted as follows: The problem (P) has a solution in a space V if (P) is well defined, if the principal operator A is linear with respect to the solution u, if A is elliptic in V, and if A is continuous in V. Each clause triggers the execution of another clause, MACSYMA functions, or LISP functions. All the results or facts obtained during the inference process are kept in a fact data base to avoid useless computations and thereby to increase the speed of the system.

In the listing in Figures 6a,b inputs are on the lines terminated with ";". In the first portion of the program the system explains its notation and then asks the user to type in the equation in a specific notation. It then asks for the boundary condition; and then it repeats the statement of the problem in symbolic form. The system then asks for the nature of the object "u", the solution. The user supplies this, and the system summarizes what it knows about u; in particular, it does not yet know the space in which u is defined and it does not know any properties of u. This process is repeated for each element appearing in the original equation. Function space and inequality properties can be given for each function.

When the problem is completely specified, the system enters a PROLOG loop. At this point the user can ask several questions about the problem; in the listing he asks the system to find out whether or not the problem posed has a solution. The system prints out everything it has proved: it finds that the problem is well defined, tests the linearity, computes the variational formulation, and the associated Sobolev space, and verifies coercivity. At the end of the manipulations the system reports that the equation has a solution in a particular Sobolev space which is $H^1(\Omega)$ in the first example and $H_0^1(\Omega) \cap L^p(\Omega)$ in the second. The total computation times for the two examples are 9 and 13 seconds, respectively.

The system operates by putting the problem in various canonical forms (i.e., canonical forms of $A\ u$ and canonical forms for the variational formulation) where it can check the criteria spelled out in certain standard theorems for the existence and uniqueness of solutions to nonlinear PDE's. All the computations are done in symbolic form, and the conditions are stored and checked in symbolic form.

5. Status and objectives of the research

At the time of this writing, the work is progressing on several fronts. New modules are being added to the system, including a system for identification of nonlinear diffusions with jumps, a system for the analysis and design of nonlinear control systems using geometric methods, together with new optimization routines, including routines for search and scheduling problems. The filtering module is being enhanced to handle vector problems more efficiently, and it is being tested on a variety of problems. The natural language interface is being redesigned to incorporate more logic (PROLOG) functions. The system, which now has some 15000 lines of code in LISP, MACSYMA, and LOGIS, has been ported on a (Symbolics) Lisp machine at INRIA.

The development of the program so far has concentrated on providing capabilities for executing specific control theoretical constructions. In the next phase of the work it will be important to bring to bear more of the techniques of expert systems to provide a systematic logical structure for the system. This appears to be the appropriate long term direction for the work.

```
EXEMPLE 1 :
*********

(c1) cogito();

Cogito : Version 4 (13/06/84)

                              n
dimension de l'espace R  :
3;

la dimension de l'espace est 3

est-ce-correct ?
oui;

equation a resoudre dans omega sous la forme "A(u)=f" :
-delta(u)+a*u=f;

conditions sur la frontiere gamma sous la forme "[B[1](u)=0,...,B[m](u)=0]" :
[diff(u,nor)=0];

le probleme a resoudre est :
          /\
a u - /__\ u = f dans omega

avec sur gamma :
 du
---- = 0
dnor

est-ce-correct ?
oui;

nature de a :
coefficient;

espace de a :
espace(1,inf);

proprietes de a :
a>1;

fonction : a
**********

    nature : coefficient

    espace : L(inf)

    proprietes : a > 1

est-ce-correct ?
oui;

nature de u :
solution;
```

```
fonction : u
**********

    nature : solution

    espace : inconnu

    proprietes : nil

est-ce-correct ?


fonction : f
**********

    nature : smembre

    espace : inconnu

    proprietes : nil

est-ce-correct ?
oui;

==>  ((solution *espace))
**** bien_defini ****

**** lineaire ****
op_principal
solution

**** formulation_variationnelle ****
op_principal
 3
====  /              /
\     [  du   dv     [
 >    I (---  ---) + I (a u v)
/     ]  dx   dx     ]
====  /    i    i    /
i = 1
/
[
I (f v)
]
/
H(1)

**** elliptique ****
op_principal
H(1)
1

**** continu **** (non developpe)
op_principal
H(1)

**** solution ****
H(1)

==>  fin
Time= 8809 msec.
```

Figure 6a. Well-posedness of a linear PDE.

```
EXEMPLE 2 :
*********

(c1) cogito();

Cogito : Version 4 (13/06/84)

                         n
dimension de l'espace R   :
2;

la dimension de l'espace est 2

est-ce-correct ?
oui;

equation a resoudre dans omega sous la forme "A(u)=f" :
u+abs(u)^(p-2)*u-sum(diff(a[i]*diff(u,x[i]),x[i]),i,1,n)=f;

conditions sur la frontiere gamma sous la forme "[B[1](u)=0,...,B[m](u)=0]" :
[u=0];

le probleme a resoudre est :
   n
  ====
  \       d      du          p - 2
-  >     --- (a  ---) + u abs(u)      + u = f dans omega
  /      dx    i dx
  ====    i       i
  i = 1

avec sur gamma :
u = 0

est-ce-correct ?
oui;

nature de u :
solution;

fonction : u
**********

    nature : solution

    espace : inconnu

    proprietes : nil

est-ce-correct ?
oui;

nature de p :
constante;

proprietes de p :
p>2;


fonction : p
**********

    nature : constante

    espace : R

    proprietes : p > 2

est-ce-correct ?
oui;

nature de a :
coefficient;

espace de a :
espace(1,inf);

proprietes de a :
a>1;
```

```
fonction : a
**********

    nature : coefficient

    espace : L(inf)

    proprietes : a > 1

est-ce-correct ?
oui;

proprietes de f :
nil;

fonction : f
**********

    nature : smembre

    espace : inconnu

    proprietes : nil

est-ce-correct ?
oui;

==>  ((solution *espace))
**** bien_defini ****

**** bien_defini **** (deja demontre)

**** non_lineaire ****
op_principal
solution



**** formulation_variationnelle ****
op_principal
 2
====  /                  /                            /
\     [      du   dv     [               p - 2        [
 >    I (a   ---  ---) + I (u abs(u)            v) + I (u v)
/     ]    i dx   dx     ]                            ]
====  /         i    i   /                            /
i = 1
/
[
I (f v)
]
/
[H(1, 0), L(p)]

**** coercif ****
op_principal
[H(1, 0), L(p)]
1

**** hemicontinu **** (non developpe)
op_principal
[H(1, 0), L(p)]

**** borne **** (non developpe)
op_principal
[H(1, 0), L(p)]

**** monotone **** (non developpe)
op_principal
[H(1, 0), L(p)]

**** solution ****
[H(1, 0), L(p)]

==>  fin
Time= 13181 msec.
```

Figure 6b. Well-posedness of a nonlinear PDE.

References

[1] P. Gloess and J. Marcovitch, *OBLOGIS: A flexible implementation of PROLOG logic and applications to DBMS interfacing andexpert systems design,* Proc. First Int. Conf. Appl. Artificial Intelligence to Engineering Problems, Southampton University, April 1986.

[2] L.S. Su, O. Ahkrif, and G.L. Blankenship, *An expert system using computer algebra for design of nonlinear control systems,* Proc. 1986 AACC, Seattle, June 1986.

[3] C. Gomez, J.P. Quadrat, and A. Sulem, *Towards an expert system in stochastic control: the Hamilton-Jacobi part,* in Lecture Notes in Control and Information Sciences, A. Bensoussan and J.L. Lions, eds., Springer-Verlag, June 1984.

[4] E.S. Armstrong, *ORACLS. A Design System for Linear Multivariable Control,* Marcel Dekker, Inc., New York, 1980.

[5] D. G. Mayne, W. T. Nye, E. Polak, P. Siegel, and T. Wuu, *DELIGHT-MIMO: An interactive, optimization-based multivariable control system design package,* Control System Magazine, vol. 2, 1982.

[6] M.K.H. Fan, W.T. Nye, and A.L. Tits, *DELIGHT.MaryLin User's Guide,* Electrical Engineering Department, University of Maryland, College Park, Feb. 1985.

[7] J.R. James, P.P. Bonissone, D.K. Frederick, and J.H. Taylor, *A retrospective view of CACE-III: Considerations in coordinating sysmbolic and numeric computation in a rule-based expert system,* preprint 1985.

[8] H.A. Spang III, *The Federated Computer-Aided control design system,* Proc. IEEE, vol. 72, 1984, pp. 1724-1731.

[9] A. J. Laub, *Numerical linear algebra aspects of control design computations,* IEEE Trans. Automatic Control, vol. AC-30, 1985, pp. 97-108.

[10] F. Delebecque, and J.P. Quadrat, *Contribution of stochastic control singular perturbation averaging and team theories to an example of large-scale systems: Management of hydropower production,* IEEE Trans. Automatic Control, vol. AC-23, 1978, pp. 209-221.

[11] P.Y. Gloess, *LOGIS User's Manual,* UTC, ERA No. 925, BP 233, 60206 COMPIEGNE CEDEX, June 1984.

[12] H. J. Kushner, *Probability Methods in Stochastic Control and for Elliptic Equations,* Academic Press, New York, 1977.

[13] J.P. Quadrat, *Analyse numerique de l'equation de Bellman stochastic,* IRIA Report, 1975.

[14] J.P. Quadrat, *Existence de solution et algorithme de resolutions numeriques de problemes stochastiques degeneres ou non,* SIAM J. Control, 1980.

[15] J.P. Quadrat and M. Viot, *Product form and optimal local feedback for multi-index Markov chains, Proc. Allerton Conf. Circuits and Systems, 1980.*

[16] J.C. Dodu, M. Goursat, A. Hertz, J.P. Quadrat, and M. Viot, *Methodes de gradient stochastic pour l'optimization des investissements dans un reseau electrique,* EDF Bulletin Serie C, 1981.

[17] A. Bensoussan, *Methode de Perturbation en Controle Optimal,* 1985, to appear.

[18] W.H. Fleming, *Control for small noise intensities,* SIAM J. Control, vol. 9, 1971.

[19] M. Hazewinkel and J.C. Willems, *Stochastic Systems: The Mathematics of Filtering and Identification and Applications,* NATO Advanced Study Institute Series, Reidel, Dordrecht, The Netherlands, 1981.

[20] S.I. Marcus, C.-H. Liu, and G.L. Blankenship, *Asymptotic expansions and Lie algebras for some nonlinear filtering problems,* IEEE Trans. Automatic Control, vol. AC-28, 1983, pp. 787-797.

[21] S.I. Marcus, *Algebraic and geometric methods in nonlinear filtering,* SIAM J. Control and Optimization, vol. 22, 1984, pp. 817-844.

[22] M. Hazewinkel, *On deformations, approximations and nonlinear filtering,* Systems and Control Letters, vol. 1, 1981, pp. 32-36.

[23] M. Hazewinkel and S.I. Marcus, *On Lie algebras and finite dimensional filtering,* Stochastics, vol. 7, 1982, pp. 29-62.

[24] J. Wei and E. Norman, *On the global representations of the solutions of linear differential equations as a product of exponentials,* Proc. Amer. Math. Soc., vol. 15, 1964, pp. 327-334.

[25] E. Pardoux and D. Talay, *Discretization and simulation of stochastic differential equations,* Publication de Mathematiques Appliquees Marseille - Toulon, No. 83-5, 1983.

[26] J.S. Baras, G.L. Blankenship, and W.E. Hopkins, Jr., *Existence, uniqueness, and asymptotic behavior of solutions to a class of Zakai equations with unbounded coefficients,* IEEE Transactions on Automatic Control, vol. AC-28, 1983, pp. 203-214.

[27] G.L. Blankenship and J.S. Baras, *Accurate evaluation of stochastic Wiener integrals with applications to scattering in random media and to nonlinear filtering,* SIAM J. Appl. Math., vol. 41, 1981, pp. 518-552.

[28] W.E. Hopkins, Jr., G.L. Blankenship, and J.S. Baras, *Accurate evaluation of conditional densities arising in the nonlinear filtering of diffusion processes,* Proc. IFIP Workshop on Recent Advances in Filtering and Optimization, W. Fleming and L. Gorostiza, eds., Springer-Verlag, New York, 1982, pp. 54-68.

[29] G.L. Blankenship, A. LaVigna, D.C. MacEnany, J.P. Quadrat, and I. Yan, *A MACSYMA/expert system for nonlinear filtering,* to appear.

Appendix: Automatically generated FORTRAN programs

belman.fortran

```
      subroutine prodyn(n1,n2,epsimp,impmax,v,ro,u,eps,nmax)
      dimension v(n1,n2),u(2,n1,n2)

c      Resolution de l equation de Bellman dans le cas ou:
c          Les parametres sont
c          L etats-temps est:  x1 x2
c          La dynamique du systeme est decrite par l operateur
c
```

```
c     Minu( (0.5 - x1)**2  (1 - x2)**2  x2**2 + (1 - x1) x1 (1 - x2) x2
c
c + 2 (1 - x2) x2 + (1 - x1)**2  x1**2 (0.5 - x2)**2  + 2 (1 - x1) x1 + u2**2  + p2 u2
c + u1**2  + p1 u1 + q2 + q1 )
c
c           ou v designe le cout optimal
c           ou pi designe sa derivee premiere par rapport a xi
c           ou qi designe sa derivee seconde par rapport a xi
c           Le probleme est statique
c           Les conditions aux limites sont:
c               x2 = 0 v= x1 x2
c               x2 = 1 v= (1 - x1) (1 - x2)
c               x1 = 0 v= x1 x2
c               x1 = 1 v= (1 - x1) (1 - x2)
c     Les nombres de points de discretisation sont: n1 n2
c               x2 = 1 correspond a i2 = n2
c               x2 = 0 correspond a i2 = 1
c               x1 = 1 correspond a i1 = n1
c               x1 = 0 correspond a i1 = 1
c     Le taux d actualisation vaut: 1
c     impmax designe le nbre maxi d iterations du systeme implicite
c     epsimp designe l erreur de convergence du systeme implicite
c     ro designe le pas de la resolution du systeme implicite
c                               par une methode iterative
c     Minimisation par la methode de Newton de l'Hamiltonien
c     L inversion de la Hessienne est faite formellement
c     nmax designe le nombre maxi d iteration de la methode de Newton
c     eps designe l erreur de convergence de la methode de Newton

      h2 = float(1)/(n2-1)
      h1 = float(1)/(n1-1)
      u2 = u(2,1,1)
      u1 = u(1,1,1)
      hih2 = h2**2
      hih1 = h1**2
      h22 = 2*h2
      h21 = 2*h1
      nm2 = n2-1
      nm1 = n1-1
      do  119   i2 = 1 , n2 , 1
      do  119   i1 = 1 , n1 , 1
      v(i1,i2) = 0.0
  119 continue
      imiter = 1
  113 continue
      erimp = 0
      do  111   i1 = 1 , n1 , 1
      x1 = h1*(i1-1)
      v(i1,n2) = 0
      v(i1,1) = 0
  111 continue
      do  109   i2 = 2 , nm2 , 1
      x2 = h2*(i2-1)
      v(n1,i2) = 0
      v(1,i2) = 0
```

```
110 continue
    do  109  i1 = 2 , nm1 , 1
    x1 = h1*(i1-1)
    q2 = (v(i1,i2+1)-2*v(i1,i2)+v(i1,i2-1))/h1h2
    q1 = (v(i1+1,i2)-2*v(i1,i2)+v(i1-1,i2))/h1h1
    p2 = (v(i1,i2+1)-v(i1,i2-1))/h22
    p1 = (v(i1+1,i2)-v(i1-1,i2))/h21
    niter = 0
    w0 = -1.0e+20
101 continue
    niter = niter+1
    if ( niter - nmax )  102 , 102 , 103
103 continue
    write(8,901)i1,i2
901 format(' newton n a pas converge', 2 i3)
    goto  104
102 continue
    u1 = -p1/2.0
    u2 = -p2/2.0
    ww = (0.5-x1)**2*(1-x2)**2*x2**2+(1-x1)*x1*(1-x2)*x2+2*(1-x2)*x2+(
   1  1-x1)**2*x1**2*(0.5-x2)**2+2*(1-x1)*x1+u2**2+p2*u2+u1**2+p1*u1+
   2  q2+q1
    er = abs(ww-w0)
    if ( er - eps )  104 , 104 , 105
105 continue
    w0 = ww
    goto  101
104 continue
    u(1,i1,i2) = u1
    u(2,i1,i2) = u2
    w0 = ww
    w0 = w0-v(i1,i2)
    vnew = ro*w0+v(i1,i2)
    v(i1,i2) = vnew
    erimp = abs(w0)+erimp
109 continue
    imiter = imiter+1
    if ( imiter - impmax )  116 , 115 , 115
116 continue
    if ( epsimp - erimp )  113 , 112 , 112
115 continue
    write(8,907)
907 format(' schema implicite n a pas converge')
112 continue
    do  117   i1 = 1 , n1 , 1
    do  117   i2 = 1 , n2 , 1
    write(8,900)i1,i2,v(i1,i2)
900 format(' v[', (i3,','), i3,']:', e14.7,'$')
    write(8,902)i1,i2,u(1,i1,i2),u(2,i1,i2)
902 format(' u[', (i3,','), i3,']:[', (e14.7,','), e14.7,']$')
117 continue
    return
    end
```

```
c
c  This is a routine that solves for the conditional estimate of
c     x at t, given the observations y up to time t.  It is written
c     by Macsyma
c
c
c  Program Variables
c
c      xinput   real(4000)
c               contains the actual x values which are to be estimated for
c               times at instants of 0.001 sec.
c
c      observ   real(4000)
c               contains the sample observations to be filtered again at
c               0.001 sec time intervals.
c
c      estim    real (4000)
c               contains the calculated estimate given the observations up to
c               that time instant.
c
      real xinput(4002), var(4002), observ(4002), estim(4002), g(10), ng
     1   (10), x0xfx, x1xfx, x2xfx, x3xfx, x4xfx
      integer i, j, k
      do  300   ii = 1 , 300 , 1
      j=1
      k=0
      size=0.001
      xinput(0)=1
      observ(0)=1
      estim(0)=1
      do  100   i = 1 , 10 , 1
         g(i)=0
100   continue
110   continue
      if(g(1) .gt. 2.5) goto 120
         call nexty(xinput,observ,1,size,j,k)
         deltay=observ(j)-observ(k)
c
         sinhg1=sinh(1.0*g(1))
         sinhg2=sinh(2.0*g(1))
         coshg1=cosh(1.0*g(1))
         coshg2=cosh(2.0*g(1))
         ng(1)=size+g(1)
c
c
         ng(2)=coshg2*deltay+coshg1*deltay-1.0*deltay+g(2)
c
c
         ng(3)=-2.0*deltay*sinhg2+2.0*deltay*sinhg1+g(3)
c
c
         ng(4)=coshg1*deltay+g(4)
c
c
         ng(5)=g(5)-1.0*deltay*sinhg1
```

```
c
c
        ng(6)=coshg2*deltay-2.0*coshg1*deltay+deltay+g(6)
c
c
        cst1=-coshg1*sinhg2*size+coshg2*sinhg1*size+2.0*coshg1*sinhg1*
    1      size-sinhg1*size+coshg1*deltay**2*sinhg2
        cst2=-coshg2*deltay**2*sinhg1-2.0*coshg1*deltay**2*sinhg1+delt
    1      ay**2*sinhg1-2.0*g(2)*deltay*sinhg1-g(3)*coshg1*deltay
        ng(7)=cst2+cst1+g(7)
c
c
        cst3=-sinhg1*sinhg2*size+sinhg1**2*size-coshg1*coshg2*size+2.0
    1      *coshg1**2*size-coshg1*size
        cst4=deltay**2*sinhg1*sinhg2-deltay**2*sinhg1**2-g(3)*deltay*s
    1      inhg1+coshg1*coshg2*deltay**2-2.0*coshg1**2*deltay**2
        cst5=coshg1*deltay**2+2.0*g(4)*coshg2*deltay-4.0*g(4)*coshg1*d
    1      eltay+2.0*g(4)*deltay+g(8)
        ng(8)=cst5+cst4+cst3
c
c
        ng(9)=0.5*coshg1*sinhg1*size-0.5*coshg1*deltay**2*sinhg1-1.0*g
    1      (4)*deltay*sinhg1+g(9)
c
c
        cst6=-g(4)*sinhg1*sinhg2*size+g(5)*coshg1*sinhg2*size+g(4)*sin
    1      hg1**2*size+g(2)*sinhg1**2*size-g(5)*coshg2*sinhg1*size
        cst7=-2.0*g(5)*coshg1*sinhg1*size+g(3)*coshg1*sinhg1*size+g(5)
    1      *sinhg1*size-g(4)*coshg1*coshg2*size+2.0*g(4)*coshg1**2*siz
    2      e
        cst8=-g(4)*coshg1*size+g(4)*deltay**2*sinhg1*sinhg2-g(5)*coshg
    1      1*deltay**2*sinhg2-deltay*sinhg2-g(4)*deltay**2*sinhg1**2
        cst9=-g(2)*deltay**2*sinhg1**2+g(5)*coshg2*deltay**2*sinhg1+2.
    1      0*g(5)*coshg1*deltay**2*sinhg1-g(3)*coshg1*deltay**2*sinhg1
    2      -g(5)*deltay**2*sinhg1
        cst10=2.0*g(2)*g(5)*deltay*sinhg1-g(3)*g(4)*deltay*sinhg1+delt
    1      ay*sinhg1+g(4)*coshg1*coshg2*deltay**2-2.0*g(4)*coshg1**2*d
    2      eltay**2
        cst11=g(4)*coshg1*deltay**2+g(4)**2*coshg2*deltay+g(3)*g(5)*co
    1      shg1*deltay-2.0*g(4)**2*coshg1*deltay+g(4)**2*deltay
        ng(10)=cst9+cst8+cst7+cst6+cst11+cst10+g(10)
c
c
      do  105   i = 1 , 10 , 1
           g(i)=ng(i)
105   continue
c
        x0xfx=x0fx(g)
        x1xfx=x1fx(g)
        x2xfx=x2fx(g)
        x3xfx=x3fx(g)
        x4xfx=x4fx(g)
        sinhg1=sinh(1.0*g(1))
        sinhg2=sinh(2.0*g(1))
        sinhg3=sinh(3.0*g(1))
      coshg1=cosh(1.0*g(1))
      coshg3=cosh(3.0*g(1))
      coshg2=cosh(2.0*g(1))
      cst12=-0.25*coshg1**2*sinhg3*x3xfx/sinhg1**2-0.083333333333333
     1   33*sinhg3*x3xfx-0.75*sinhg1*x3xfx+0.25*coshg1*coshg3*x3xfx/
     2   sinhg1+1.5*coshg1**2*x3xfx/sinhg1
      cst13=-coshg1*x3xfx/sinhg1+0.08333333333333333*coshg1**3*coshg
     1   3*x3xfx/sinhg1**3-0.75*coshg1**4*x3xfx/sinhg1**3+0.66666666
     2   66666667*coshg1**3*x3xfx/sinhg1**3-0.5*g(5)*coshg1*sinhg3*x
     3   2xfx/sinhg1**2
      cst14=0.5*coshg1*sinhg3*x2xfx/sinhg1**2-g(6)*coshg1*sinhg2*x2x
     1   fx/sinhg1-g(2)*coshg1*sinhg2*x2xfx/sinhg1+0.5*g(3)*coshg1**
     2   2*sinhg2*x2xfx/sinhg1**2+0.5*g(3)*sinhg2*x2xfx
      cst15=0.25*g(5)*coshg3*x2xfx/sinhg1-0.25*coshg3*x2xfx/sinhg1-g
     1   (3)*coshg1*coshg2*x2xfx/sinhg1+2.25*g(5)*coshg1*x2xfx/sinhg
     2   1-2.25*coshg1*x2xfx/sinhg1
      cst16=-g(5)*x2xfx/sinhg1+x2xfx/sinhg1+0.5*g(6)*coshg1**2*coshg
     1   2*x2xfx/sinhg1**2+0.5*g(2)*coshg1**2*coshg2*x2xfx/sinhg1**2
     2   +0.5*g(6)*coshg1**2*x2xfx/sinhg1**2
      cst17=-0.5*g(2)*coshg1**2*x2xfx/sinhg1**2+0.25*g(5)*coshg1**2*
     1   coshg3*x2xfx/sinhg1**3-0.25*coshg1**2*coshg3*x2xfx/sinhg1**
     2   3-2.25*g(5)*coshg1**3*x2xfx/sinhg1**3+2.25*coshg1**3*x2xfx/
     3   sinhg1**3
      cst18=2.0*g(5)*coshg1**2*x2xfx/sinhg1**3-2.0*coshg1**2*x2xfx/s
     1   inhg1**3+0.5*g(6)*coshg2*x2xfx+0.5*g(2)*coshg2*x2xfx-0.5*g(
     2   6)*x2xfx
      cst19=0.5*g(2)*x2xfx+0.5*coshg1*sinhg3*x1xfx/sinhg1-0.25*g(5)*
     1   *2*sinhg3*x1xfx/sinhg1**2+0.5*g(5)*sinhg3*x1xfx/sinhg1**2-0
     2   .25*sinhg3*x1xfx/sinhg1**2
      cst20=-g(5)*g(6)*sinhg2*x1xfx/sinhg1+g(6)*sinhg2*x1xfx/sinhg1-
     1   g(2)*g(5)*sinhg2*x1xfx/sinhg1+g(2)*sinhg2*x1xfx/sinhg1+g(3)
     2   *g(5)*coshg1*sinhg2*x1xfx/sinhg1**2
      cst21=-g(3)*coshg1*sinhg2*x1xfx/sinhg1**2+g(8)*sinhg1*x1xfx-2.
     1   0*g(4)*g(6)*sinhg1*x1xfx-g(3)*g(5)*coshg2*x1xfx/sinhg1+g(3)
     2   *coshg2*x1xfx/sinhg1
      cst22=-g(8)*coshg1**2*x1xfx/sinhg1+2.0*g(4)*g(6)*coshg1**2*x1x
     1   fx/sinhg1+0.75*g(5)**2*x1xfx/sinhg1-1.5*g(5)*x1xfx/sinhg1+0
     2   .75*x1xfx/sinhg1
      cst23=-0.25*coshg1**2*coshg3*x1xfx/sinhg1**2+g(5)*g(6)*coshg1*
     1   coshg2*x1xfx/sinhg1**2-g(6)*coshg1*coshg2*x1xfx/sinhg1**2+g
     2   (2)*g(5)*coshg1*coshg2*x1xfx/sinhg1**2-g(2)*coshg1*coshg2*x
     3   1xfx/sinhg1**2
      cst24=2.25*coshg1**3*x1xfx/sinhg1**2-2.0*coshg1**2*x1xfx/sinhg
     1   1**2+g(5)*g(6)*coshg1*x1xfx/sinhg1**2-g(6)*coshg1*x1xfx/sin
     2   hg1**2-g(2)*g(5)*coshg1*x1xfx/sinhg1**2
      cst25=g(2)*coshg1*x1xfx/sinhg1**2+0.25*g(5)**2*coshg1*coshg3*x
     1   1xfx/sinhg1**3-0.5*g(5)*coshg1*coshg3*x1xfx/sinhg1**3+0.25*
     2   coshg1*coshg3*x1xfx/sinhg1**3-2.25*g(5)**2*coshg1**2*x1xfx/
     3   sinhg1**3
      cst26=4.5*g(5)*coshg1**2*x1xfx/sinhg1**3-2.25*coshg1**2*x1xfx/
     1   sinhg1**3+2.0*g(5)**2*coshg1*x1xfx/sinhg1**3-4.0*g(5)*coshg
     2   1*x1xfx/sinhg1**3+2.0*coshg1*x1xfx/sinhg1**3
      cst27=-0.25*coshg3*x1xfx-2.25*coshg1*x1xfx+x1xfx+0.25*g(5)*sin
     1   hg3*x0xfx/sinhg1-0.25*sinhg3*x0xfx/sinhg1
      cst28=-0.5*g(3)*coshg1*sinhg2*x0xfx/sinhg1+0.5*g(3)*g(5)**2*si
     1      nhg2*x0xfx/sinhg1**2-g(3)*g(5)*sinhg2*x0xfx/sinhg1**2+0.5*g
     2      (3)*sinhg2*x0xfx/sinhg1**2+0.5*g(6)*sinhg2*x0xfx
        cst29=0.5*g(2)*sinhg2*x0xfx-0.5*g(6)*coshg1*coshg2*x0xfx/sinhg
     1      1-0.5*g(2)*coshg1*coshg2*x0xfx/sinhg1-g(5)*g(8)*coshg1*x0xf
     2      x/sinhg1+g(8)*coshg1*x0xfx/sinhg1
        cst30=2.0*g(4)*g(5)*g(6)*coshg1*x0xfx/sinhg1-2.0*g(4)*g(6)*cos
     1      hg1*x0xfx/sinhg1-0.5*g(6)*coshg1*x0xfx/sinhg1+0.5*g(2)*cosh
     2      g1*x0xfx/sinhg1-0.25*g(5)*coshg1*coshg3*x0xfx/sinhg1**2
        cst31=0.25*coshg1*coshg3*x0xfx/sinhg1**2+0.5*g(5)**2*g(6)*cosh
     1      g2*x0xfx/sinhg1**2-g(5)*g(6)*coshg2*x0xfx/sinhg1**2+0.5*g(6
     2      )*coshg2*x0xfx/sinhg1**2+0.5*g(2)*g(5)**2*coshg2*x0xfx/sinh
     3      g1**2
        cst32=-g(2)*g(5)*coshg2*x0xfx/sinhg1**2+0.5*g(2)*coshg2*x0xfx/
     1      sinhg1**2+2.25*g(5)*coshg1**2*x0xfx/sinhg1**2-2.25*coshg1**
     2      2*x0xfx/sinhg1**2-2.0*g(5)*coshg1*x0xfx/sinhg1**2
        cst33=2.0*coshg1*x0xfx/sinhg1**2+0.5*g(5)**2*g(6)*x0xfx/sinhg1
     1      **2-g(5)*g(6)*x0xfx/sinhg1**2+0.5*g(6)*x0xfx/sinhg1**2-0.5*
     2      g(2)*g(5)**2*x0xfx/sinhg1**2
        cst34=g(2)*g(5)*x0xfx/sinhg1**2-0.5*g(2)*x0xfx/sinhg1**2+0.083
     1      33333333333333*g(5)**3*coshg3*x0xfx/sinhg1**3-0.25*g(5)**2*
     2      coshg3*x0xfx/sinhg1**3+0.25*g(5)*coshg3*x0xfx/sinhg1**3
        cst35=-0.08333333333333333*coshg3*x0xfx/sinhg1**3-0.75*g(5)**3
     1      *coshg1*x0xfx/sinhg1**3+2.25*g(5)**2*coshg1*x0xfx/sinhg1**3
     2      -2.25*g(5)*coshg1*x0xfx/sinhg1**3+0.75*coshg1*x0xfx/sinhg1*
     3      *3
        cst36=0.6666666666666667*g(5)**3*x0xfx/sinhg1**3-2.0*g(5)**2*x
     1      0xfx/sinhg1**3+2.0*g(5)*x0xfx/sinhg1**3-0.6666666666666667*
     2      x0xfx/sinhg1**3+0.5*g(3)*coshg2*x0xfx
        cst37=g(10)*x0xfx-g(4)*g(8)*x0xfx+g(7)*x0xfx+g(4)**2*g(6)*x0xf
     1      x-0.75*g(5)*x0xfx
        u1=-0.5*g(3)*x0xfx+0.75*x0xfx+cst37+cst36+cst35+cst34+cst33+cs
     1      t32+cst31+cst30+cst29+cst28+cst27+cst26+cst25+cst24+cst23+c
     2      st22+cst21+cst20+cst19+cst18+cst17+cst16+cst15+cst14+cst13+
     3      cst12
c
        cst38=-0.25*coshg1**2*sinhg3*x4xfx/sinhg1**2-0.083333333333333
     1      33*sinhg3*x4xfx-0.75*sinhg1*x4xfx+0.25*coshg1*coshg3*x4xfx/
     2      sinhg1+1.5*coshg1**2*x4xfx/sinhg1
        cst39=-coshg1*x4xfx/sinhg1+0.08333333333333333*coshg1**3*coshg
     1      3*x4xfx/sinhg1**3-0.75*coshg1**4*x4xfx/sinhg1**3+0.66666666
     2      66666667*coshg1**3*x4xfx/sinhg1**3-0.5*g(5)*coshg1*sinhg3*x
     3      3xfx/sinhg1**2
        cst40=0.5*coshg1*sinhg3*x3xfx/sinhg1**2-g(6)*coshg1*sinhg2*x3x
     1      fx/sinhg1-g(2)*coshg1*sinhg2*x3xfx/sinhg1+0.5*g(3)*coshg1**
     2      2*sinhg2*x3xfx/sinhg1**2+0.5*g(3)*sinhg2*x3xfx
        cst41=0.25*g(5)*coshg3*x3xfx/sinhg1-0.25*coshg3*x3xfx/sinhg1-g
     1      (3)*coshg1*coshg2*x3xfx/sinhg1+2.25*g(5)*coshg1*x3xfx/sinhg
     2      1-2.25*coshg1*x3xfx/sinhg1
        cst42=-g(5)*x3xfx/sinhg1+x3xfx/sinhg1+0.5*g(6)*coshg1**2*coshg
     1      2*x3xfx/sinhg1**2+0.5*g(2)*coshg1**2*coshg2*x3xfx/sinhg1**2
     2      +0.5*g(6)*coshg1**2*x3xfx/sinhg1**2
        cst43=-0.5*g(2)*coshg1**2*x3xfx/sinhg1**2+0.25*g(5)*coshg1**2*
     1      coshg3*x3xfx/sinhg1**3-0.25*coshg1**2*coshg3*x3xfx/sinhg1**
     2      3-2.25*g(5)*coshg1**3*x3xfx/sinhg1**3+2.25*coshg1**3*x3xfx/
     3      sinhg1**3
```

```
      cst44=2.0*g(5)*coshg1**2*x3xfx/sinhg1**3-2.0*coshg1**2*x3xfx/s
     1   inhg1**3+0.5*g(6)*coshg2*x3xfx+0.5*g(2)*coshg2*x3xfx-0.5*g(
     2   6)*x3xfx
      cst45=0.5*g(2)*x3xfx+0.5*coshg1*sinhg3*x2xfx/sinhg1-0.25*g(5)*
     1   *2*sinhg3*x2xfx/sinhg1**2+0.5*g(5)*sinhg3*x2xfx/sinhg1**2-0
     2   .25*sinhg3*x2xfx/sinhg1**2
      cst46=-g(5)*g(6)*sinhg2*x2xfx/sinhg1+g(6)*sinhg2*x2xfx/sinhg1-
     1   g(2)*g(5)*sinhg2*x2xfx/sinhg1+g(2)*sinhg2*x2xfx/sinhg1+g(3)
     2   *g(5)*coshg1*sinhg2*x2xfx/sinhg1**2
      cst47=-g(3)*coshg1*sinhg2*x2xfx/sinhg1**2+g(8)*sinhg1*x2xfx-2.
     1   0*g(4)*g(6)*sinhg1*x2xfx-g(3)*g(5)*coshg2*x2xfx/sinhg1+g(3)
     2   *coshg2*x2xfx/sinhg1
      cst48=-g(8)*coshg1**2*x2xfx/sinhg1+2.0*g(4)*g(6)*coshg1**2*x2x
     1   fx/sinhg1+0.75*g(5)**2*x2xfx/sinhg1-1.5*g(5)*x2xfx/sinhg1+0
     2   .75*x2xfx/sinhg1
      cst49=-0.25*coshg1**2*coshg3*x2xfx/sinhg1**2+g(5)*g(6)*coshg1*
     1   coshg2*x2xfx/sinhg1**2-g(6)*coshg1*coshg2*x2xfx/sinhg1**2+g
     2   (2)*g(5)*coshg1*coshg2*x2xfx/sinhg1**2-g(2)*coshg1*coshg2*x
     3   2xfx/sinhg1**2
      cst50=2.25*coshg1**3*x2xfx/sinhg1**2-2.0*coshg1**2*x2xfx/sinhg
     1   1**2+g(5)*g(6)*coshg1*x2xfx/sinhg1**2-g(6)*coshg1*x2xfx/sin
     2   hg1**2-g(2)*g(5)*coshg1*x2xfx/sinhg1**2
      cst51=g(2)*coshg1*x2xfx/sinhg1**2+0.25*g(5)**2*coshg1*coshg3*x
     1   2xfx/sinhg1**3-0.5*g(5)*coshg1*coshg3*x2xfx/sinhg1**3+0.25*
     2   coshg1*coshg3*x2xfx/sinhg1**3-2.25*g(5)**2*coshg1**2*x2xfx/
     3   sinhg1**3
      cst52=4.5*g(5)*coshg1**2*x2xfx/sinhg1**3-2.25*coshg1**2*x2xfx/
     1   sinhg1**3+2.0*g(5)**2*coshg1*x2xfx/sinhg1**3-4.0*g(5)*coshg
     2   1*x2xfx/sinhg1**3+2.0*coshg1*x2xfx/sinhg1**3
      cst53=-0.25*coshg3*x2xfx-2.25*coshg1*x2xfx+x2xfx+0.25*g(5)*sin
     1   hg3*x1xfx/sinhg1-0.25*sinhg3*x1xfx/sinhg1
      cst54=-0.5*g(3)*coshg1*sinhg2*x1xfx/sinhg1+0.5*g(3)*g(5)**2*si
     1   nhg2*x1xfx/sinhg1**2-g(3)*g(5)*sinhg2*x1xfx/sinhg1**2+0.5*g
     2   (3)*sinhg2*x1xfx/sinhg1**2+0.5*g(6)*sinhg2*x1xfx
      cst55=0.5*g(2)*sinhg2*x1xfx-0.5*g(6)*coshg1*coshg2*x1xfx/sinhg
     1   1-0.5*g(2)*coshg1*coshg2*x1xfx/sinhg1-g(5)*g(8)*coshg1*x1xf
     2   x/sinhg1+g(8)*coshg1*x1xfx/sinhg1
      cst56=2.0*g(4)*g(5)*g(6)*coshg1*x1xfx/sinhg1-2.0*g(4)*g(6)*cos
     1   hg1*x1xfx/sinhg1-0.5*g(6)*coshg1*x1xfx/sinhg1+0.5*g(2)*cosh
     2   g1*x1xfx/sinhg1-0.25*g(5)*coshg1*coshg3*x1xfx/sinhg1**2
      cst57=0.25*coshg1*coshg3*x1xfx/sinhg1**2+0.5*g(5)**2*g(6)*cosh
     1   g2*x1xfx/sinhg1**2-g(5)*g(6)*coshg2*x1xfx/sinhg1**2+0.5*g(6
     2   )*coshg2*x1xfx/sinhg1**2+0.5*g(2)*g(5)**2*coshg2*x1xfx/sinh
     3   g1**2
      cst58=-g(2)*g(5)*coshg2*x1xfx/sinhg1**2+0.5*g(2)*coshg2*x1xfx/
     1   sinhg1**2+2.25*g(5)*coshg1**2*x1xfx/sinhg1**2-2.25*coshg1**
     2   2*x1xfx/sinhg1**2-2.0*g(5)*coshg1*x1xfx/sinhg1**2
      cst59=2.0*coshg1*x1xfx/sinhg1**2+0.5*g(5)**2*g(6)*x1xfx/sinhg1
     1   **2-g(5)*g(6)*x1xfx/sinhg1**2+0.5*g(6)*x1xfx/sinhg1**2-0.5*
     2   g(2)*g(5)**2*x1xfx/sinhg1**2
      cst60=g(2)*g(5)*x1xfx/sinhg1**2-0.5*g(2)*x1xfx/sinhg1**2+0.083
     1   333333333333333*g(5)**3*coshg3*x1xfx/sinhg1**3-0.25*g(5)**2*
     2   coshg3*x1xfx/sinhg1**3+0.25*g(5)*coshg3*x1xfx/sinhg1**3
      cst61=-0.0833333333333333*coshg3*x1xfx/sinhg1**3-0.75*g(5)**3
     1   *coshg1*x1xfx/sinhg1**3+2.25*g(5)**2*coshg1*x1xfx/sinhg1**3
```

```
    2      -2.25*g(5)*coshg1*x1xfx/sinhg1**3+0.75*coshg1*x1xfx/sinhg1*
    3      *3
         cst62=0.6666666666666667*g(5)**3*x1xfx/sinhg1**3-2.0*g(5)**2*x
    1      1xfx/sinhg1**3+2.0*g(5)*x1xfx/sinhg1**3-0.6666666666666667*
    2      x1xfx/sinhg1**3+0.5*g(3)*coshg2*x1xfx
         cst63=g(10)*x1xfx-g(4)*g(8)*x1xfx+g(7)*x1xfx+g(4)**2*g(6)*x1xf
    1      x-0.75*g(5)*x1xfx
         u1x=-0.5*g(3)*x1xfx+0.75*x1xfx+cst63+cst62+cst61+cst60+cst59+c
    1      st58+cst57+cst56+cst55+cst54+cst53+cst52+cst51+cst50+cst49+
    2      cst48+cst47+cst46+cst45+cst44+cst43+cst42+cst41+cst40+cst39
    3      +cst38
c
         estim(j)=(u1x-u1*x1xfx/x0xfx)/x0xfx+x1xfx/x0xfx
         var(j)=(xinput(j)-estim(j))**2+var(j)
         k=j
         j=j+1
      go to 110
120   continue
300   continue
      do 400  i = 1 , 4000 , 1
      var(i)=var(i)/300.0
      y=i*size
      print*,y,var(i)
400   continue
      stop
      end

      real function x0fx(g)
      real g(1)
      x0fx=1/sqrt(cosh(g(1)))
      return
      end

      real function x1fx(g)
      real g(1)
      x1fx=(1-g(5))*cosh(g(1))**((-3.0)/2.0)
      return
      end

      real function x2fx(g)
      real g(1)
      x2fx=cosh(g(1))**((-5.0)/2.0)*(cosh(g(1))*sinh(g(1))+g(5)**2-2*g(5
     1   )+1)
      return
      end

      real function x3fx(g)
      real g(1)
      x3fx=(1-g(5))*cosh(g(1))**((-7.0)/2.0)*(3*cosh(g(1))*sinh(g(1))+g(
     1   5)**2-2*g(5)+1)
      return
      end

      real function x4fx(g)
      real g(1)
```

```
 x4fx=cosh(g(1))**((-9.0)/2.0)*(3*cosh(g(1))**2*sinh(g(1))**2+6*g(5
1   )**2*cosh(g(1))*sinh(g(1))-12*g(5)*cosh(g(1))*sinh(g(1))+6*cosh
2   (g(1))*sinh(g(1))+g(5)**4-4*g(5)**3+6*g(5)**2-4*g(5)+1)
 return
 end

 subroutine nexty(xinput,observ,e,size,j,k)
 real xinput(1), observ(1), e, size
 integer j, k
 xinput(j)=grand(0)*sqrt(size)+xinput(k)
 observ(j)=(e*xinput(j)**2+xinput(j))*size+grand(0)*sqrt(size)+obse
1   rv(k)
 return
 end
```